Interfacce hardware

Lezione 11 di Sistemi dedicati

Docente: Giuseppe Scollo

Università di Catania
Dipartimento di Matematica e Informatica
Corso di Laurea Magistrale in Informatica, AA 2017-18

Indice

argomenti della lezione

di che si tratta:

- l'interfaccia hardware di coprocessore
- funzioni tipiche delle interfacce hardware
- struttura di un'interfaccia hardware di coprocessore
- indirizzamento dei dati
- multiplazione
- masking
- progetto del controllo
- controllo gerarchico
- modello di programmazione
  - mappa degli indirizzi
  - insieme delle istruzioni
- esempio: un caso di accelerazione hardware
  - decisioni di progetto
  - interfaccia Avalon e modello di programmazione

funzioni, struttura e progetto di interfacce hardware

seminario di Salvatore Mameli (PDF)

modello di programmazione

modello di programmazione = progetto del controllo + progetto dei dati

il *modello di programmazione*, cioè la vista software di un modulo hardware, include:

- una collezione delle locazioni di memoria usate dal modulo hardware custom, e
- una definizione dei comandi (o istruzioni) accettati dal modulo

seguono alcune considerazioni sull'impatto che questi due tipi di decisioni di progetto hanno sul progetto del driver software del modulo hardware custom

mappa degli indirizzi

la *mappa degli indirizzi* riflette l'organizzazione di elementi di memoria del modulo hardware accessibili al software in lettura e in scrittura; il suo progetto dovrebbe procedere dal punto di vista del progettista del software piuttosto che dell'hardware, perciò:

- un dato indirizzo di memoria dovrebbe sempre riferirsi allo stesso registro hardware, indipendentemente dal tipo di operazione, lettura o scrittura, su di esso
- per default tutti i registri mappati in memoria dovrebbero essere accessibili in lettura e scrittura; in alcuni casi sono giustificati registri solo in lettura, e.g. registri di stato dell'hardware o dati di segnali campionati; tuttavia, sono molto rari i casi che giustificano l'uso di un registro solo in scrittura
- la mappa degli indirizzi dovrebbe rispettare l'allineamento del processore; e.g., estrarre i bit 5–12 da una parola di 32 bit è più complicato che estrarre il secondo byte della stessa parola

insieme delle istruzioni

il progetto di un buon insieme di istruzioni è un problema difficile, che si pone al progettista in termini di bilanciamento tra flessibilità ed efficienza

il problema dipende molto dalla funzione del modulo hardware custom

ecco alcune generiche linee-guida per il progetto:

- si possono distinguere tre classi di istruzioni: comandi one-time, comandi on-off, configurazioni; la loro miscela influisce sul comportamento generale del modulo hardware e dovrebbe essere mirata a minimizzare l'entità dell'interazione di controllo fra driver software e modulo hardware
- progettare la sincronizzazione tra software e hardware a più livelli di astrazione, cioè non solo al livello del trasferimento di dati ma anche a quello algoritmico
- un altro problema di sincronizzazione si pone quando più utenti software condividono uno stesso modulo hardware; lo si può risolvere serializzando l'uso del coprocessore o realizzando un commmutatore di contesto nel modulo hardware
- infine, la progettazione del reset va considerata con attenzione: un esempio di debolezza del progetto si ha quando per inizializzare un modulo hardware è necessario un reset generale di sistema; è sensato definire una o più istruzioni per l'inizializzazione e per il reset del modulo

esempio: decisioni di progetto per un caso di accelerazione hardware

in una recente esercitazione è stata presentata una realizzazione software del calcolo del delay di una traiettoria di Collatz di dato inizio

realizzazioni hardware della stessa funzione sono state oggetto di precedenti esperienze di laboratorio

e.g. la terza esperienza di laboratorio ne produce una descrizione in VHDL

le misure di prestazione condotte sulla realizzazione software mostrano che essa assorbe quasi tutto il tempo di esecuzione del programma

problema: accelerare l'esecuzione del programma usando la realizzazione hardware della funzione suddetta

una prima alternativa da valutare: integrare la funzione hardware come istruzione custom o come coprocessore mappato in memoria?

la seconda opzione appare migliore, per almeno due ragioni:

- la prima opzione è *bloccante*
- il volume di dati trasferiti in ogni interazione è molto piccolo

altre decisioni di progetto dipendono da questa prima decisione, come segue

interfaccia Avalon e modello di programmazione per il caso in esempio

la descrizione VHDL del circuito di calcolo della funzione va incorporata in un componente dotato di interfacce Avalon per i segnali di Clock, Reset e di Avalon MM Slave, sì da ricevere il dato iniziale da un'operazione di scrittura e fornire il risultato in risposta a un'operazione di lettura

- trasferimenti di dati *multiciclo* sono possibili grazie al segnale Avalon `waitrequest`, impostato dallo slave per differire la risposta a una richiesta di lettura o scrittura per un numero arbitrario di cicli

indirizzamento del coprocessore: poiché le operazioni di scrittura (del dato iniziale) e lettura (del risultato) avvengono in tempi diversi e hanno la stessa dimensione del dato, un solo indirizzo è sufficiente

- per semplicità conviene usare i segnali Avalon a 32 bit `writedata`, `readdata` nell'interfaccia hardware per questo indirizzo, con conversione interna a 16 bit per le corrispondenti porte interne di I/O del circuito di calcolo della funzione

driver software: si possono definire due macro e una funzione per l'interfaccia software di accesso al bus: `DC_RESET(d)`, `DC_START(d,x0)`, `unsigned int delay(d)`, dove `d` è l'indirizzo assegnato al coprocessore

- queste idee di progetto saranno sviluppate nella prossima esercitazione

riferimenti

letture raccomandate:

- Schaumont, Ch. 12, Sect. 12.1-12.3.1, 12.4

per ulteriore consultazione:

- Schaumont, Ch. 12, Sect. 12.3.2
- Avalon® Interface Specifications, Ch. 1-3, MNL-AVABUSREF, Intel Corp., 2017.05.08